ALGORITHMIQUE · CHAPITRE 8 · FONDATIONS

Passer du texte aux données du calcul

Distinguez lecture, conversion, validation et affichage. Transformez un texte numérique en valeur et repérez les erreurs de syntaxe, de calcul et de données.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Séparer texte reçu, nombre converti et résultat affiché.
- Distinguer une conversion réussie et une donnée autorisée.
- Reconnaître erreur de syntaxe, erreur d'exécution et erreur logique.

Prérequis. Connaître les types, les comparaisons et les limites des nombres.

Le texte "12" peut-il servir directement de quantité ?

Une saisie au clavier ou un champ de fichier fournit souvent du texte. Voir les caractères 1 et 2 ne signifie pas encore disposer du nombre douze. Pour calculer le prix de douze objets à 250 centimes, nous devons convertir ce texte, puis vérifier que la quantité est autorisée.

Lecture, conversion, validation et sortie

Une **chaîne de caractères** est une valeur textuelle, délimitée ici par des guillemets. Lire acquiert une donnée ; convertir change sa représentation ou son type ; valider vérifie les règles du problème ; afficher rend une information visible. Ces étapes ont des rôles différents. Pour une quantité entière non négative, convertir le texte "-2" réussit, mais le résultat ne respecte pas le domaine autorisé.

Étapes du traitement

Étape	Exemple	Question
Lecture	Texte "12"	Quelle donnée a été reçue ?
Conversion	Nombre 12	Le texte décrit-il un nombre admissible pour la conversion ?
Validation	Entier et non négatif	Cette valeur est-elle autorisée par le contrat ?
Calcul	12 × 250	Quelles unités utilise-t-on ?
Affichage	3000 centimes	Comment communiquer le résultat ?

Python `input("Quantity: ")` renvoie le texte saisi. MATLAB `input('Quantity: ', 's')` lit également du texte. Les exemples suivants fixent l'entrée pour être reproductibles ; vous pourrez remplacer cette ligne par une lecture. Le pseudo-code du cours utilise `READ` pour recevoir une entrée ; `parse_integer` désigne dans notre pseudo-code la conversion d'un texte décimal valide en entier et signale une erreur sinon.

Convertir une donnée connue

Python `int(text)` lit ici un entier décimal. MATLAB `str2double(text)` produit un nombre flottant ; obtenir un nombre ne garantit donc pas qu'il soit entier. Les deux versions reçoivent le texte valide "12" ; la validation générale sera construite après les conditions.

Pseudo-code

```
// example.pseudo
// convert_quantity
// Input is fixed here; only a valid decimal integer is supplied.
text <- "12"
quantity <- parse_integer(text)
total <- quantity * 250
PRINT total
```

Python

```python
# convert_quantity.py
text = "12"
quantity = int(text)
total = quantity * 250
print(total)
```

MATLAB

```matlab
% convert_quantity.m
text = "12";
quantity = str2double(text);
total = quantity * 250;
disp(total);
```

Le résultat attendu est 3000 centimes. Le texte n'a pas été multiplié : c'est la valeur convertie qui entre dans le calcul. En Python, `"12" * 2` répète le texte et donne "1212" ; `12 * 2` donne le nombre 24. Le type détermine le sens de l'opération.

Reconnaître les familles d'erreurs

Une **erreur de syntaxe** empêche de lire le programme, par exemple une parenthèse non fermée. Une **erreur d'exécution** survient pendant son traitement, comme une conversion invalide en Python. Une **erreur logique** laisse le programme s'exécuter mais produit un résultat incorrect, par exemple en additionnant quantité et prix au lieu de les multiplier. Une donnée invalide peut également être signalée par une valeur spéciale.

Avec le texte "twelve", Python `int` signale une exception `ValueError` ; MATLAB `str2double` renvoie `NaN`, qui signifie ici qu'aucun nombre n'a été obtenu. `NaN` ne se teste pas par égalité à lui-même : MATLAB fournit `isnan`. Ce sont deux comportements différents à prendre en compte avant de poursuivre le calcul.

Les exemples numériques utilisent le point décimal dans le code : `12.5`. Python `int("12.5")` échoue ; `float("12.5")` lit un flottant. MATLAB `str2double("12.5")` donne 12,5, qui n'est pas une quantité entière. Ne supposez pas que la virgule française soit automatiquement un séparateur décimal reconnu.

Essayer succès et échec séparément

Dans chaque langage, essayez les textes "0", "-2", "12.5" et "twelve" séparément : un échec Python arrête le script. Notez si la conversion réussit, puis si la valeur satisfait le contrat d'une quantité entière non négative. Seul "0" satisfait les deux dans cette liste. Gardez le message d'erreur et le numéro de ligne pour retrouver l'instruction concernée.

Définir avant de valider

▼ Convertir "-2" en nombre suffit-il à accepter une commande ?

Non. La conversion réussit mais le contrat interdit les quantités négatives. La validation doit précéder le calcul métier.

▼ Afficher un résultat équivaut-il à le renvoyer ?

Non. Afficher permet de le lire à l'écran. Une fonction pourra plus tard renvoyer une valeur à un autre calcul, sans nécessairement afficher quoi que ce soit.

Référence MATLAB : conversion de texte avec str2double.

À retenir - fiche mémo

Conversion

Lire le texte "12" comme le nombre 12 précède son utilisation numérique.

Validation

Une conversion réussie ne garantit pas que la valeur appartienne au domaine autorisé.

Diagnostic

Distinguez syntaxe illisible, échec d'exécution et résultat logiquement incorrect.

Pseudo-code

```
// example.pseudo
// convert_quantity
text <- "12"
quantity <- parse_integer(text)
PRINT quantity * 250
```

Point de vigilance. Le texte "-2" se convertit, mais ne satisfait pas une quantité entière non négative.

Consulter la cheatsheet : Fondations · **Télécharger la fiche du thème en PDF**

Version web : https://www.clearcode.fr/courses/programming/algorithmic/foundations/input-output